Machine Learning and Mass Estimation Methods for Ground-Based Aircraft Climb Prediction

Richard Alligier, David Gianazza, Nicolas Durand

Machine Learning and Mass Estimation Methods for Ground-based Aircraft Climb Prediction

R. Alligier, D. Gianazza, N. Durand
ENAC, MAIAA, F-31055 Toulouse, France
Univ. de Toulouse, IRIT/APO, F-31400 Toulouse, France

***Abstract*—In this paper, we apply Machine Learning methods to improve the aircraft climb prediction in the context of ground-based applications. Mass is a key parameter for climb prediction. As it is considered a competitive parameter by many airlines, it is currently not available to ground-based trajectory predictors. Consequently, most predictors today use a reference mass that may be different from the actual aircraft mass. In previous papers, we have introduced a least square method to estimate the mass from past trajectory points, using the physical model of the aircraft. Another mass estimation method, based on an adaptive mechanism, has also been proposed by Schultz et. al.**

We now introduce a new approach, where the mass is considered as the response variable of a prediction model that is learned from a set of example trajectories. This Machine Learning approach is compared with the results obtained when using the BADA (Base of Aircraft Data) reference mass or the two state-of-the-art mass estimation methods. In these experiments, 9 different aircraft types are considered.

When compared with the baseline method (resp. the mass estimation methods), the Machine Learning approach reduces the RMSE (Root Mean Square Error) on the predicted altitude by at least 58 % (resp. 27 %) when assuming the speed profile to be known, and by at least 29 % (resp. 17 %) when using the BADA speed profile except for the aircraft types E145 and F100. For these types, the observed speed profile is far from the BADA speed profile.

***Index Terms*—aircraft trajectory prediction, mass estimation, BADA, Machine Learning**

INTRODUCTION

Aircraft trajectory prediction has always been a key issue for many on-board and ground-based applications in Air Transportation. It is even more true since the current Air Traffic Management and Control (ATM/ATC) system is shifting towards trajectory-based operations within the framework of the European SESAR program ([1]) and its U.S. counterpart NextGen ([2]).

As we are now in an era of global networks, where data flows between flying aircraft and ground-based control systems, one could think that ground-based trajectory prediction is no longer necessary: accurate on-board trajectory predictions could be downloaded directly to the ground systems. Although this last statement is actually true, we are still in great need of accurate ground-based trajectory predictors. Some of the most recent algorithms designed to solve ATM/ATC problems do require to test a large number of alternative trajectories and it would be impractical to download them all from the aircraft. As an example of such algorithms, in [3] an iterative quasi-Newton method is used to find trajectories for departing aircraft, minimizing the noise annoyance. Another example is [4] where Monte Carlo simulations are used to estimate the risk of conflict between trajectories in a stochastic environment. Some of the automated tools currently being developped for ATM/ATC can detect and solve conflicts between trajectories (see [5] for a review). These algorithms may use Mixed Integer Programming ([6]), Genetic Algorithms ([7], [8]), Ant Colonies ([9]), or Differential Evolution or Particle Swarm Optimization ([10]) to find optimal solutions to air traffic conflicts.

To be efficient, all these methods require a fast and accurate trajectory prediction, and the capability to test a large number of "what-if" trajectories. Such requirements forbid the sole use of on-board trajectory prediction, which is certainly the most accurate, but is not sufficient for these most promising applications.

Most trajectory predictors rely on a point-mass model to describe the aircraft dynamics. The aircraft is simply modeled as a point with a mass, and the second Newton's law is applied to relate the forces acting on the aircraft to the inertial acceleration of its center of mass. Such a model is formulated as a set of differential algebraic equations that must be integrated over a time interval in order to predict the successive aircraft positions, knowing the aircraft initial state (mass, current thrust setting, position, velocity, bank angle, etc.), atmospheric conditions (wind, temperature), and aircraft intent (thrust profile, speed profile, route).

Unfortunately, the data that is currently available to ground-based systems for trajectory prediction purposes is of fairly poor quality. The speed intent and aircraft mass, being considered competitive parameters by many airline operators, are not transmitted to ground systems. The actual thrust setting of the engines (nominal, reduced, or other, depending on the throttle's position) is unknown. There are uncertainties or noise in the Weather and Radar data. The problem of unknown parameters such as the mass, thrust law, and target speeds, is of particular importance when predicting the aircraft climb. Figure 1 illustrates the climb prediction problem, when using a physical model of the aircraft dynamics.

Some studies ([11], [12], [13]) detail the potential benefits that would be provided by additional or more accurate input data. In other works, the aircraft intent is formalized through the definition of an Aircraft Intent Description Language ([14], [15]) that could be used in air-ground data links to transmit some useful data to ground-based applications. All the necessary data required to predict aircraft trajectories might become available to ground systems someday. In the meantime, we

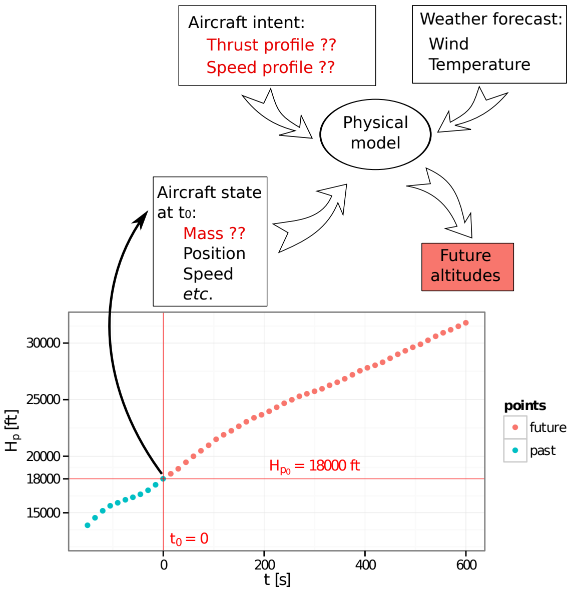

Figure 1: The ground-based aircraft climb prediction problem

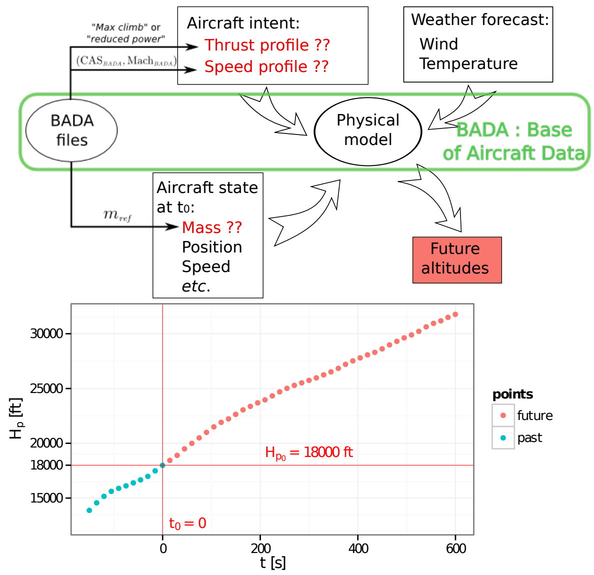

Figure 2: Baseline method : the BADA prediction of the future aircraft climb

propose to learn some of the unknown parameters of the point-mass model from the data that is already available today, typically from the observed radar tracks of past and current flights.

In this paper, we apply Machine Learning techniques to learn the aircraft mass. We show how our method improves the climb prediction when compared with the baseline method (see Figure 2) that uses the reference mass from the Eurocontrol Base of Aircraft Data (BADA). We also compare our Machine Learning approach to two other mass estimation methods ([16], [17]) that rely solely on the physical model of the aircraft dynamics to estimate the mass from the past trajectory points.

The rest of this paper is organized as follows: Section I gives some background on the estimation of aircraft model parameters and highlights the differences between mass prediction (using Machine Learning) as we introduce it in this paper, and mass estimation (using the physical model). Section II details the data used in this study. Section III presents some useful Machine Learning notions that help understanding the methodology applied in our work. The application of Machine Learning techniques to our mass prediction problem is described in section IV, and the results are shown and discussed in section V, before the conclusion.

I. Background, mass prediction vs. estimation

A. *Estimation of physical model parameters*

Focusing on the aircraft climb, and considering a physical model of the aircraft dynamics, we are interested in this paper in estimating some key parameters for climb performance using the past trajectory points. This approach, where some unknown parameters are adjusted by fitting the model to the observed past trajectory, is not new. The past publications following this path ([18], [19], [20], [21], [17], [22], [23], [24]) propose several methods, with different choices for the adjusted parameter (mass, or thrust, for example), the modeled variable that is fitted on past observations (rate of climb, energy rate), and the algorithm that is applied (stochastic method, adaptive mechanism, least squares, etc).

In [21] Lymperopoulos, Lygeros, and Lecchini model the aircraft mass and the wind encountered during climb as sources of uncertainty. These stochastic variables are sampled from chosen distributions to produce random simulated trajectories. Each trajectory is then weighted according to its probability to give the aircraft positions measured just after takeoff. The uncertainty on the future aircraft positions is reduced by selecting the parameters of highest probability after a number of measurements. The method is tested in a simulation environment. In [20], Slater introduces an adaptive mechanism improving the trajectory prediction by dynamically adjusting the modeled thrust. The aircraft mass is assumed to be equal to a standard reference mass for the chosen aircraft type. The results presented in [20] show significant improvements in the climb prediction accuracy for simulated data, and much fewer improvements when applied to a few examples using real trajectories. Other works propose to adjust the mass instead of the thrust.

Among the publications dealing with mass estimation, let us cite [18], where Warren and Ebrahimi propose an *equivalent weight* as a workaround to use a point-mass model without knowing the actual aircraft mass. Nominal thrust and drag profiles are assumed. The equivalent mass is found by minimizing the gap between the computed and observed vertical rates. A second study ([19]) raises doubts about the reliability of the vertical rate for this purpose, and suggests to use the energy rate instead. The proposed method is tested on simulated trajectories only. In more recent works, Schultz, Thipphavong, and Erzberger ([17]) introduce an adaptive mechanism where the modeled mass is adjusted by fitting the modeled energy rate with the observed energy rate. This adaptive method provides good results on simulated traffic and this method has also been successfully applied on actual radar data ([25], [26]).

In [22], [23], we use a Quasi-Newton algorithm (BFGS) combined to a mass estimation method to learn the thrust profile minimizing the error between the modeled and observed energy rate. This method has been tested on two months of real data, showing good results. Concerning the mass estimation method, we showed that, when using the BADA[1] model of the forces (or a similar model), the aircraft mass can be estimated at any past point of the trajectory by solving a polynomial equation, knowing the thrust setting at this point. When using several points, and assuming a constant mass over the whole trajectory segment, the mass can be estimated by minimizing the quadratic error on the energy rate. In our latest papers ([16], [27]), we introduce a variant of this mass estimation method, taking the fuel burn into account, and compare it with the adaptive method of Schultz *et. al.*. In the current paper, we propose a completely different approach, where the aircraft mass is predicted by a model learned from examples.

B. Mass prediction vs. mass estimation

Both the adaptive ([25]) and least square ([22], [23]) methods evoked in previous subsection I-A rely solely on the physical model to estimate the mass from past trajectory points. The mass is adjusted so that the modeled power fits the energy rate observed on the past points, assuming the thrust profile to be known. This mass estimation approach is illustrated on the left part of Figure 3.

The Machine Learning approach we introduce in this paper makes use of additional data found in a database of trajectory examples. The idea is to learn a prediction model from the examples, as illustrated on the right part of Figure 3. Instead of directly adjusting the mass on the past trajectory points, we adjust a prediction model on a set of examples. Once the model is calibrated, it can be used to predict the mass on fresh trajectory inputs. The methodological issues concerning model selection, parameter tuning, and performance assessment are briefly presented in section III. For now, let us just say that, in the Machine Learning approach, a set of examples $(y_i, x_i)_{1 \leqslant i \leqslant n}$ is used to build a model which relates the predicted variable y to some explanatory variables x. In our case, the predicted variable y is the aircraft mass m. Unfortunately – and this is the crux of our problem – the actual mass is not available in our data.

In order to build examples that can be used by Machine Learning algorithms, we propose, for each example trajectory, to adjust a modeled mass so that the modeled power fits the observed energy rate as best as possible on the "future" points. Here, the terms "past" and "future" refer to the fact that the aircraft altitude is respectively below or above a reference pressure altitude H_{p_0}, assuming we want to predict the successive altitudes above H_{p_0} when the climbing aircraft crosses altitude H_{p_0}. The modeled mass is adjusted using the least square method introduced in [22], [23].

In other words, we propose to replace the actual mass m, missing in our data, by an adjusted mass $\hat{m}_{\mathit{future}}$ that gives the best possible fit of the energy rate on our examples, assuming a *max climb* thrust setting. This mass $\hat{m}_{\mathit{future}}$ is the y output variable of the prediction model learned by the Machine Learning methods. The explanatory variables x, are computed from the "past" data that is available when the aircraft crosses H_{p_0}.

The Radar and Weather data, as well as the construction of our examples, are described in more details in section II.

C. Applying our method in actual operations

The proposed method consists in learning a model that can predict the aircraft mass, given Radar and Weather data inputs, or any other relevant additionnal inputs (e.g. flight plan). This model is learned on a dataset of example trajectories. Once learned, it can be used for predicting the aircraft mass on fresh input trajectories. The predicted mass can then be used as input to the physics-based model that is already used in operations, in order to produce an accurate trajectory prediction.

When applying such a Machine Learning method in operations, one must first collect trajectory data, build a training set, and tune the model. This should be done for every aircraft type for which there is sufficient training data. Tuning one model per aircraft is no more an issue than for the standard BADA model, for which there is also one model per aircraft type or group of similar aircraft. Note however that, when using a Machine Learning approach, the performance of the tuned model highly depends on the quality of the collected data. For more accuracy, the training datasets should be specific to each airport or terminal area where we intend to apply our tuned model.

Note that in that respect, our approach is much easier to put in operations than purely data-driven methods as we still use the physics-based model. Purely data-driven approach rely on a statistical model to predict directly the altitude. This requires to tune a specific model for each mode of operation (e.g. climb at constant rate, or at constant calibrated airspeed, *etc.*). This means we need sufficient data for each aircraft type and each mode of operation, and for every airport where we intend to use such a model. In our case, we use the Machine Learning approach only to learn one of the input parameters – here the mass – of the physics-based model. This model is already in operation and remains valid whatever the chosen mode of operation. Furthermore, for the aircraft types or airports for which there is not enough data of sufficient quality to learn a model predicting the mass, we can easily revert to the mass estimated solely from the past trajectory points, or to the reference BADA mass, while still using the physics-based model.

Airport and airline procedures might change over years, as well as the performances of the engines equiping the aircraft.

[1] BADA: the Eurocontrol Base of Aircraft DAta

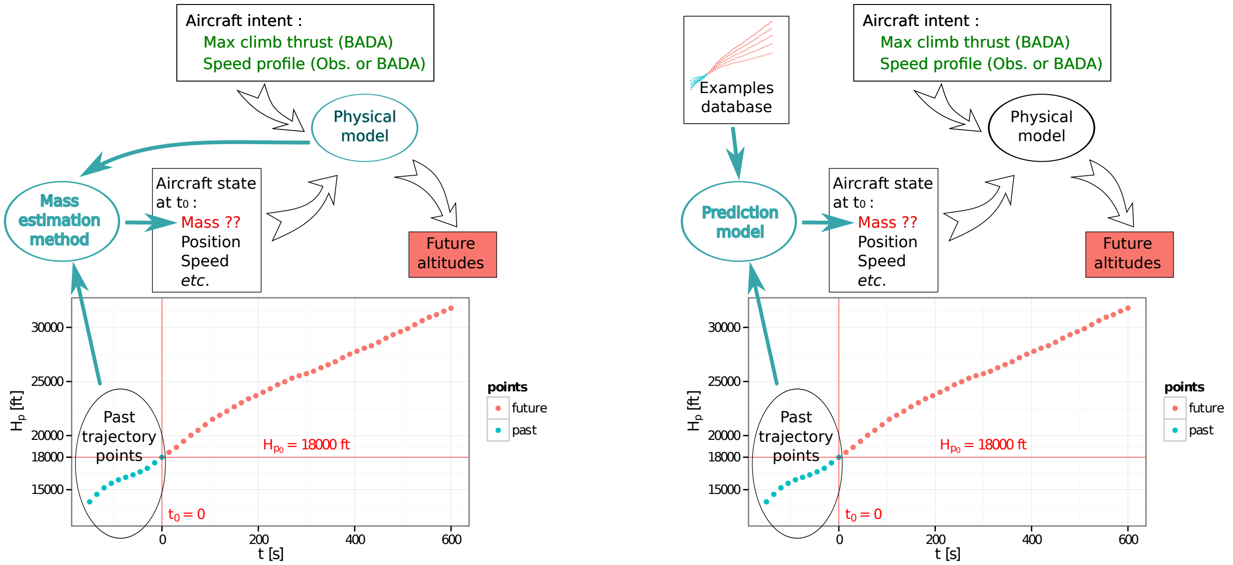

Figure 3: Mass estimation (left) vs. mass prediction (right)

These changes obviously impact the performance of the tuned model. To address this issue, one can monitor the model performance over time and tune it again when it becomes less performant.

II. Data used in this Study

A. Data Pre-processing

Recorded radar tracks from Paris Air Traffic Control Center are used in this study. This raw data is made of one position report every 1 to 3 seconds, over two months (July 2006, and January 2007). In addition, the wind and temperature data from Météo France are available at various isobar altitudes over the same two months.

The raw Mode-C altitude[2] has a precision of 100 feet. Raw trajectories are smoothed using splines. Basic trajectory data is made of the following fields: aircraft position (X,Y in a projection plane, or latitude and longitude in WGS84), ground velocity vector $V_g = (V_x, V_y)$, smoothed altitude (H_p, in feet above isobar 1,013.25 hPa), rate of climb or descent $\frac{dH_p}{dt}$. The wind $W = (W_x, W_y)$ and temperature T at every trajectory point are interpolated from the weather datagrid. The temperature differential ΔT is computed at each point of the trajectory.

Using the position, velocity and wind data, we compute the true air speed V_a. The successive velocity vectors allow us to compute the trajectory curvature at each point. The aircraft bank angle is then derived from true airspeed and the curvature of the air trajectory.

Along with these variables derived from the Mode-C radar data and the weather data, we have access to some variables in the flight plan like the Requested Flight Level for instance.

With the weather datagrid, we have also computed the temperature differential ΔT(weather grid) and the wind along W_{along}(weather grid) at each altitude of the grid. This is done by using the V_{aXY}, the time, the latitude and the longitude of the considered point. W_{along} is the wind along the true air speed in the horizontal plane V_{aXY}.

All the computed variables are summarized in table I.

B. Filtering Climb Segments

Our dataset comprises all flights departing from Paris-Orly (LFPO) or Paris-Charles de Gaulle Airport (LFPG). Needless to say, this approach can be replicated to other airports.

The trajectories are filtered so as to keep only the climb segments. An additional 80 seconds is clipped from the beginning and end of each segment so as to remove climb/cruise or cruise/climb transitions.

C. Building the Sets of Examples

The climb segments are sampled every 15 seconds. From these sampled segments, we build examples (or patterns) containing exactly 51 points. In these examples, the first 11 points (past trajectory) are used to predict the mass. The remaining points (future trajectory) are used to compute the error between the predicted and actual trajectory. In this study, we use two different datasets.

1) A small A320 dataset: In order to compare the different Machine Learning methods, we use a small set of examples noted $A320_{small}$. This set of examples is built using climbing segments of aircraft of type A320. The climbing segments are sampled in order to have the 11^{th} point always[3] at 18,000ft.

[2] This altitude is directly derived from the air pressure measured by the aircraft. It is the height in feet above isobar 1013.25 hPa.

[3] Using the smoothed altitude $H_p(t)$, we search for the time t_0 such that $H_p(t_0)$ = 18,000 ft. Once this time is found, we sample 10 points before and 40 points after.

variables	description
H_p	geopotential pressure altitude
V_g	Ground Speed
V_a	True Air Speed
V_{aXY}	True Air Speed in the (X,Y) plane
d_{air}	distance flown w.r.t. the air
d_{ground}	distance flown w.r.t. the ground
ΔT	temperature differential (cf. [28])
W	wind
W_{along}	wind along V_{aXY}
W_{across}	wind across V_{aXY}
W_Z	vertical wind
θ_c	drift angle
CAS	Calibrated Air Speed
Mach	Mach number
$1/r_{\text{sol}}$	curvature w.r.t. the ground
$1/r_{\text{air}}$	curvature w.r.t. the air
ϕ	bank angle
$e = V_a \frac{dV_a}{dt} + g_0 \frac{T}{T-\Delta T} \frac{dH_p}{dt}$	specific energy rate
$ew = e + \overrightarrow{W}.\overrightarrow{V_a}$	specific energy rate corrected from the wind effect
$\hat{m}_{LS}$	estimated mass from past points using least square method [16]
e_{LS}	root mean square error obtained on the past points using the least square method
$\hat{m}_{AD}$	estimated mass from past points using adaptive method [17]
ΔT (weather grid)	temperature differential on a grid of different H_p
W_{along} (weather grid)	wind along V_{aXY} on a grid of different H_p
RFL	Requested Flight Level
Speed	requested speed
distance	distance between airports
AO	aircraft operator
DEP	departing airport
ARR	arrival airport

Table I: This table summarizes the variables available in our study.

Thus, from one sampled climb segment, we build only one example. The resulting set of examples, denoted $A320_{small}$ in the following, contains 4,939 examples. It is only used to choose a machine learning method.

2) A larger dataset with 9 *aircraft types and various altitudes:* This larger dataset is used to compare the selected Machine Learning method to the baseline BADA predictor and to the two other methods using estimated masses (adaptive, or least squares), on 9 different aircraft types and with various altitudes for the "current" point.

The sampled climbing segments and the examples are built in a different way. The raw climb segments are smoothed and sampled every 15 seconds, starting at the first point. With each sampled climb segment, we build as much examples containing 51 successive points as we can. The 10 first points of each example are considered as the "past" points. The 11^{th} point is the "current" point, and the next 40 points are the "future" points used to evaluate the prediction made using the 11 first points.

As a consequence, the 11^{th} point is not always at 18,000ft. As we are mostly interested in altitude prediction in the *en-route* airspace, we only keep the patterns with the 11^{th} point at an altitude above 15,000ft[4] for the B744 aircraft type and above 18,000ft for all the other aircraft types.

The multiple examples extracted from a same trajectory might share many points. Consequently, when splitting our set into a training set and a test set, the results would be biased if we put some of these examples in the training set and the others in the test set. We have been very careful not to do that. The training and test sets are built by choosing randomly among the trajectories, not the examples they contain.

We have considered 9 aircraft types and we have built one examples dataset for each aircraft type. Table II shows the size of the different datasets. The selected aircraft types are very different: for example, the E145 is a short haul aircraft with a 18,500 kg reference mass while the B744 (Boeing 747-400) is a long haul aircraft with a 285,700 kg reference mass.

type	number of climbing segments	number of examples
A319	1863	15702
A320	5729	65514
A321	1866	21789
A332	1475	28629
B737	344	2178
B744	350	2750
B772	910	8525
E145	851	8310
F100	660	7430

Table II: Size of the different sets. Only the climbing segments generating at least one example in our final examples set are counted here. In our data, no flight has more than one climbing segment generating examples.

D. Estimation of the mass to be predicted

The actual mass is not available in our radar data set. Thus, as explained in section I-B, we have used the least square method proposed in [16] on the 41 future points of the trajectories. This method estimates a mass sequence corresponding to a sequence of trajectory points. This mass sequence takes into account the fuel consumption. It minimizes the sum of the squared differences between the observed specific energy rate and the computed specific power (see [16] for details).

Let us denote $\hat{m}_{11,future}$ the first mass of this sequence, that is the mass at the "current" point (numbered 11 in our examples). This estimated mass $\hat{m}_{11,future}$ is the output variable y of the prediction model we want to learn from examples. To estimate this mass for each of our example trajectories, we need to make some hypotheses concerning the thrust settings, which are not available in our data. We assume a standard BADA *max climb* thrust, during all climb.

As a consequence, the estimated mass might be quite different from the actual one, especially for aircraft climbing at reduced power. This difference is not of crucial importance, however, as there is an infinity of couples $(mass, thrust\_profile)$ that give exactly the same trajectory. Intuitively, a heavy aircraft with maximum climb thrust is

[4] The chosen minimum flight level for the Boeing 747-400 is lower than for the other aircraft types because we wanted to have enough examples in our dataset.

equivalent to a lighter aircraft with reduced thrust. Although it might not be realistic, the modeled mass can be adjusted so as to give an accurate prediction of the energy rate. Knowing the energy rate and the speed intent, one can predict the future altitudes of the aircraft.

E. Approximation of the example trajectories when using the estimated mass $\hat{m}_{11,future}$

For verification purposes, let us assess the accuracy of the trajectory computed using the estimated mass on our set of examples. To do this, we compute the "future" trajectories using the speed profile $V_a = {V_a}^{(obs)}(t)$ and the mass $\hat{m}_{11,future}$, and observe the error in altitude.

time range	mean	stdev	mean abs	rmse	max abs
$t > 0$	31.7	144	111	147	1407
$t = 600$ s	-63.1	121	105	137	817

Table III: Statistics, in feet, on the difference between the computed and observed altitudes $\left(H_p^{(pred)}(\hat{m}_{11,future}) - H_p^{(obs)}\right)$ at different time ranges for the examples set $A320_{small}$.

Looking at table III, we see that the differences between the predicted altitudes and the observed ones are limited[5]. There remains an incompressible error, though, which might be due to the fact that some aircraft might not actually follow a constant *max climb* thrust law, nor a constant reduced power climb. They might switch from one to the other during the climb. Learning the thrust settings is not the subject of the current paper, and has already been investigated in [23], and we shall assume a constant *max climb* thrust in the rest of this work.

Another possible source of error is the estimation of the true airspeed V_a, which relies on the observed ground speed and the wind forecast. Due to the uncertainties in the weather forecast, and the possible observation errors in the ground speed, the "observed" airspeed profile might lack accuracy.

In any case, table III gives us an order of magnitude of the best possible approximation of the "future" trajectory that we can achieve with our data and assumptions, when using the estimated mass $\hat{m}_{11,future}$. This approximation error is computed on our set of examples $A320_{small}$, for verification purposes. It is not representative of the prediction error that will be made when considering fresh trajectories.

Let us now see how we can apply Machine Learning methods to build a model that predicts $\hat{m}_{11,future}$, and that will allow us to predict new trajectories.

III. Machine Learning

This section presents some useful Machine Learning notions. We want to predict a variable y, here the aircraft mass m of a given trajectory, from a vector of explanatory variables x, which in our case is the data extracted from the past trajectory points and the weather forecast. This is typically a regression problem. Naively said, we want to learn a function f such that $y = f(x)$ for all (x, y) drawn from the distribution (X, Y). Actually, such a function does not exist, in general. For instance, if two ordered pairs (x, y_1) and (x, y_2) can be drawn with $y_1 \neq y_2$, $f(x)$ cannot be equal to y_1 and y_2 at the same time.

A way to solve this issue is to use a real-valued *loss function* L. This function is defined by the user of the function f. The value $L(f(x), y)$ models a cost for the specific use of f when (x, y) is drawn. With this definition, the user wants a function f minimizing the expected loss $R(f)$ defined by equation (1). The value $R(f)$ is also called the *expected risk*.

$$R(f) = E_{(X,Y)}\left[L\left(f(X), Y\right)\right] \tag{1}$$

However, the main issue in order to choose a function f minimizing $R(f)$ is that we do not know the joint distribution (X, Y). We only have a set of examples of this distribution.

A. Learning from examples

Let us consider a set of n examples $S = (x_i, y_i)_{1 \leqslant i \leqslant n}$ coming from independent draws of the same joint distribution (X, Y). We can define the *empirical risk* $R_{empirical}$ by the equation below:

$$R_{empirical}(f, S) = \frac{1}{|S|} \sum_{(x,y) \in S} L\left(f(x), y\right). \tag{2}$$

Assuming that the values $(L(f(x), y))_{(x,y) \in S}$ are independent draws from the same law with a finite mean and variance, we can apply the law of large numbers giving us that $R_{empirical}(f, S)$ converges to $R(f)$ as $|S|$ approaches $+\infty$.

Thereby, the *empirical risk* is closely related to the *expected risk*. So, if we have to select f among a set of functions F minimizing $R(f)$, using a set of examples S, we select f minimizing $R_{empirical}(f, S)$. This principle is called the *principle of empirical risk minimization*.

Unfortunately, choosing f minimizing $R_{empirical}(f, S)$ will not always give us f minimizing $R(f)$. Actually, it depends on the "size"[6] of F and the number of examples $|S|$ ([29], [30]). The smaller is F and the larger is $|S|$, the more the *principle of empirical risk minimization* is relevant. When these conditions are not satisfied, the selected f will probably have a high $R(f)$ despite a low $R_{empirical}(f, S)$. In this case, the function f is *overfitting* the examples S.

These general considerations above have practical consequences on the use of Machine Learning. Let us denote f_S the function in F minimizing $R_{empirical}(., S)$. The *expected risk* using f_S is given by $R(f_S)$. We use the *principle of empirical risk minimization*. As stated above, some conditions are required for this principle to be relevant. Concerning the size of the set of examples S: the larger, the better. Concerning the size of F, there is a tradeoff. The larger F is, the smaller $\underset{f \in F}{min}\ R(f)$ is. However, the larger F is, the larger the gap between $R(f_S)$ and $\underset{f \in F}{min}\ R(f)$ becomes. This is often referred to as the *bias-variance tradeoff*.

[5] Especially when compared with the first line of table IX.

[6] The "size" of F refers here to the complexity of the candidate models contained in F, and hence to their capability to adjust to complex data. As an example, if F is a set of polynomial functions, we can define the "size" of F as the highest degree of the functions contained in F. In classification problems, the "size" of F can be formalized as the Vapnik-Chervonenkis dimension.

B. Accuracy Estimation

In this subsection, we want to estimate the accuracy obtained using a Machine Learning algorithm $\mathcal{A}$. Let us denote $\mathcal{A}[T]$ the prediction model found by algorithm $\mathcal{A}$ when minimizing $R_{empirical}(., S)^7$, considering a set of examples S.

The *empirical risk* $R_{empirical}(\mathcal{A}[S], S)$ is not a suitable estimation of $R(\mathcal{A}[S])$: the law of large numbers does not apply here because the predictor $\mathcal{A}[S]$ is neither fixed nor independent from the set of examples S.

One way to handle this is to split the set of examples S into two independent subsets: a *training set* S_T and another set S_V that is used to estimate the *expected risk* of $\mathcal{A}[S_T]$, the model learned on the training set S_T. For that purpose, one can compute the holdout validation error Err_{val} as defined by the equation below:

$$Err_{val}(\mathcal{A}, S_T, S_V) = R_{empirical}(\mathcal{A}[S_T], S_V). \tag{3}$$

Cross-validation is an other popular method that can be used to estimate the *expected risk* obtained with a given learning algorithm. In a k-fold cross-validation method, the set of examples S is partitioned into k folds $(S_i)_{1 \leqslant i \leqslant k}$. Let us denote $S_{-i} = S \backslash S_i$. In this method, k trainings are performed in order to obtain the k predictors $\mathcal{A}[S_{-i}]$. The mean of the holdout validation errors is computed, giving us the cross-validation estimation below:

$$CV(\mathcal{A}, S) = \sum_{i=1}^{k} \frac{|S_i|}{|S|} Err_{val}(\mathcal{A}, S_{-i}, S_i). \tag{4}$$

This method is more computationally expensive than the holdout method but the cross-validation is more accurate than the holdout method ([31]). In our experiments, the folds were stratified. This technique is said to give more accurate estimates ([32]).

The accuracy estimation has basically two purposes: *i*) model selection in which we select the "best" model using accuracy measurements and *ii*) model assessment in which we estimate the accuracy of the selected model. For model selection, the set S_V in $Err_{val}(\mathcal{A}, S_T, S_V)$ is called *validation set* whereas in model assessment this set is called *testing set*.

C. Hyperparameter Tuning

Some learning algorithms have hyperparameters. These hyperparameters λ are the parameters of the learning algorithm $\mathcal{A}_\lambda$. These parameters cannot be adjusted using the *empirical risk* because most of the hyperparameters are directly or indirectly related to the size of F. Thus, if the *empirical risk* was used, the selected hyperparameters would always be the ones associated to the largest F.

These hyperparameters allow us to control the size of F in order to deal with the *bias-variance tradeoff*. These hyperparameters can be tuned using the holdout method on a *validation set* for accuracy estimation. In order to find λ minimizing the accuracy estimation we have used a grid search which consists in an exhaustive search on a predefined set of hyperparameters. The algorithm 1 obtained is a learning algorithm without any hyperparameters. In this algorithm, 20% of the *training set* is held out as a *validation set*.

```
function TuneGrid($\mathcal{A}_\lambda$,$grid$)[$T$]
    $(T_T, T_V) \leftarrow split_{(80\%,20\%)}(T)$
    $\lambda^* \leftarrow \underset{\lambda \in grid}{argmin}\ Err_{val}(\mathcal{A}_\lambda, T_T, T_V)$
    return $\mathcal{A}_{\lambda^*}[T]$
end function
```

Algorithm 1: Hyperparameters tuning for an algorithm $\mathcal{A}_\lambda$ and a set of examples T (training set).

IV. Applying Machine Learning to our Problem

A. Different Sets of Variables

We want to find f such that $\hat{m}_{11,future} = f(x)$, with x the information available when the prediction is computed. The choice of explanatory variables x is of considerable importance to the performance of the prediction model.

In this work, the candidate explanatory variables x are classified in five groups of variables depending on their provenance and their type. These groups are described in table IV. The r ("radar") group contains 297 variables extracted from the available Radar data, for the 11 past points of each trajectory. The $\hat{m}$ ("mass") group contains 3 numerical variables. Two of them are masses estimated using the adaptive ([25]) and least square ([22], [23]) methods. e_{LS} is the root mean squared error of the energy rate prediction obtained with the least square method on the past points of the trajectory. The w ("weather") group contains 20 numerical variables. This group contains ΔT (weather grid) and W_{along} (weather grid) computed on the last point of the past trajectory. These quantities are computed at the 10 different altitudes of the weather grid. Finally, the p ("flight plan") group contains 3 numerical variables and group c contains 3 categorical variables, also extracted from the flight plan.

Several combinations of these groups are tested, as specified in table IV: r, $\hat{m}r$, $\hat{m}rw$, $\hat{m}prw$ and $c\hat{m}prw$. This last group contains categorical variables that can be handled straightfully by the Gradient Boosting Machine (GBM) algorithm, but not by the other Machine Learning methods that we used.

B. Machine Learning Algorithms

Five different classical Machine Learning algorithms were tested in this study. These algorithms optimize the risk given by a quadratic loss $L(\hat{y}, y) = (\hat{y} - y)^2$. A multiple linear regression on the k variables selected by a *forward-selection* MLR-FS$_k$ ([33], [34]), a Ridge regression Ridge$_\lambda$ ([35]) with λ the penalty hyperparameter, and a principal component regression PCR$_k$ ([36]) on the k principal components were tested. In these three methods, the obtained model is a linear combination of the explanatory variables. However, the obtained models are different because these algorithms have different ways to control the set of functions F through their hyperparameters. A single-layer neural network NNet$_{(n,\lambda)}$

[7] Actually, depending on the nature of the minimization problem and chosen algorithm, this predictor $\mathcal{A}[S]$ might not be the global optimum for $R_{empirical}(., S)$, especially if the underlying optimization problem is handled by local optimization methods.

variables	set of variables				
	r	$\hat{m}r$	$\hat{m}rw$	$\hat{m}prw$	$c\hat{m}prw$
H_p $\frac{dH_p}{dt}$ $\frac{d^2H_p}{dt^2}$ V_g $\frac{dV_g}{dt}$ V_a $\frac{dV_a}{dt}$ $V_{a\,XY}$ $d_{\text{air}} - d_{\text{air}\,11}$ $d_{\text{ground}} - d_{\text{ground}\,11}$ ΔT W W_{along} $\frac{dW_{\text{along}}}{dt}$ W_{across} W_Z $\frac{dW_Z}{dt}$ θ_c *CAS* $\frac{dCAS}{dt}$ *Mach* $\frac{dMach}{dt}$ $1/r_{\text{sol}}$ $1/r_{\text{air}}$ ϕ *e* *ew*	✓	✓	✓	✓	✓
$\hat{m}_{LS}$ e_{LS} $\hat{m}_{AD}$		✓	✓	✓	✓
ΔT (weather grid) W_{along} (weather grid)			✓	✓	✓
RFL *Speed* *distance*				✓	✓
AO *DEP* *ARR*					✓

Table IV: This table summarizes the different sets of variables used by the Machine Learning algorithms.

([37]) was tested with a weight decay λ and a number of n hidden units. A stochastic gradient boosting tree algorithm GBM$_{(m,J,\nu)}$ ([38]) was tested with m the boosting iterations, J the interaction depth and ν the shrinkage parameter. With this method the obtained model is a sum of m regression trees. As opposed to the other methods, this method can easily handle categorical variables without any prior encoding. The hyperparameters grids for these algorithms are presented in table V.

method	hyperparameter grid
MLR-FS$_k$	$k = [\![2; min(120, n_{var})]\!]$
Ridge$_\lambda$	$\lambda = 10^{[\![-5;1]\!]} \cup 0.5 \times 10^{[\![-5;0]\!]}$
PCR$_k$	$k = [\![2; min(120, n_{var})]\!]$
NNet$_{(n,\lambda)}$	$n = \{2, 3, 4, 5, 6\}$ $\lambda = \{0.1, 0.2, 0.3, 0.4, 0.5\}$
GBM$_{(m,J,\nu)}$	$m = \{1000, 1500, 2000\}$ $J = \{3, 5, 10, 15\}$ $\nu = \{0.001, 0.0025, 0.005, 0.01, 0.025, 0.05\}$

Table V: Grid of hyperparameters used in our experiments.

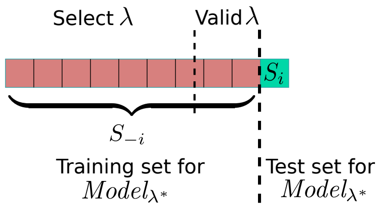
Select λ
Valid λ
S_i
S_{-i}
Training set for $Model_{\lambda^*}$
Test set for $Model_{\lambda^*}$

Figure 4: Cross-validation for model assessment, with an embedded holdout validation for hyperparameter tuning.

V. Results and Discussion

All the statistics presented in this section are computed using a stratified 10-fold cross-validation embedding the hyperparameter selection. Figure 4 illustrates how the data is partitionned, denoting λ the hyperparameter vector. Our set of examples S is partitioned in 10 folds $(S_i)_{1\leqslant i\leqslant 10}$. The hyperparameters used to learn from S_{-i} are selected using 20% of the fold S_{-i} as a *validation set*. The model learned with these hyperparameters on S_{-i} is then used to predict the mass on the *test set* S_i. Obviously, the intersection of the *training set* S_{-i} and the *test set* S_i is empty: they do not share any example. Overall, our set of predicted values (masses or altitudes) is the concatenation of the ten $TuneGrid(\mathcal{A}_\lambda, grid)[S_{-i}]\,(S_i)$ (see algorithm 1). Therefore, all the statistics presented in this section are computed on test sets. Each large set of examples S corresponding to one of the 9 aircraft types is split in 10 folds $(S_i)_{1\leqslant i\leqslant 10}$, randomly choosing the climb segments. Thus, all the examples generated by one climbing segment are only in one fold S_i. This guarantees that the examples in fold S_i are independent from the ones in $S_{-i} = S \backslash S_i$.

A. Prediction of the Mass $\hat{m}_{11,future}$

The results obtained with the Machine Learning algorithm on the set $A320_{small}$ are reported in table VI. The results obtained with the BADA reference mass m_{ref}, as well as the masses $\hat{m}_{LS}$ and $\hat{m}_{Ad}$ estimated with the least square and adaptive methods, are also stated in this table as a baseline. Looking at the 6^{th} column showing the root mean squared error (RMSE), we can see that all linear models have about the same performance. Equally, NNet and GBM perform similarly, with a slight advantage for the latter. For all methods, the more variables we have, the more accurate the prediction is. However, the error on the mass is not significantly reduced by adding the group of variables w ("wind") to the set $\hat{m}r$. The greater error reduction is obtained by adding the group $\hat{m}$ ("mass") to the "radar" set r, especially for the linear models. This was expected, as these estimated masses are highly correlated with $\hat{m}_{11,future}$, with a correlation coefficient above 0.94. In comparison, all the variables in set r have a correlation coefficient with $\hat{m}_{11,future}$ below 0.61. However, one has to keep in mind that these correlation coefficients are computed taking the variables one by one. In a regression context, all these variables are used altogether and as stated in [39]: variables that are useless by themselves can be useful together.

Among the different machine learning methods, the best results for the $A320_{small}$ dataset are obtained with the GBM method, with the variables $c\hat{m}prw$. Throughout the rest of

set	method	mean	stdev	mean abs	rmse	max abs
-	m_{ref}	0.366	7.56	5.93	7.57	31.7
-	$\hat{m}_{LS}$	2.46	3.26	2.94	4.08	34.9
-	$\hat{m}_{Ad}$	1.27	3.28	2.71	3.52	21.7
r	MLR-FS	0.0691	2.65	1.88	2.65	42.5
r	Ridge	0.0739	2.66	1.88	2.66	42.9
r	PCR	0.079	2.81	2.04	2.81	44
r	NNet	0.0493	2.31	1.68	2.31	32.3
r	GBM	0.0747	2.28	1.69	2.28	19.6
$\hat{m}r$	MLR-FS	0.062	2.43	1.75	2.43	36
$\hat{m}r$	Ridge	0.0623	2.43	1.75	2.43	37.2
$\hat{m}r$	PCR	0.0652	2.43	1.75	2.43	36.7
$\hat{m}r$	NNet	0.0501	2.23	1.65	2.23	22.5
$\hat{m}r$	GBM	0.0537	2.22	1.65	2.22	23.9
$\hat{m}rw$	MLR-FS	0.0706	2.42	1.74	2.42	36.3
$\hat{m}rw$	Ridge	0.0605	2.43	1.74	2.43	37.5
$\hat{m}rw$	PCR	0.064	2.42	1.74	2.42	37.2
$\hat{m}rw$	NNet	0.0435	2.21	1.63	2.21	24.5
$\hat{m}rw$	GBM	0.0655	2.21	1.64	2.21	23.2
$\hat{m}prw$	MLR-FS	0.0643	2.37	1.7	2.37	35.2
$\hat{m}prw$	Ridge	0.0589	2.38	1.7	2.38	36.9
$\hat{m}prw$	PCR	0.063	2.38	1.71	2.38	36.9
$\hat{m}prw$	NNet	0.0392	2.17	1.59	2.17	25.8
$\hat{m}prw$	GBM	0.0548	2.13	1.56	2.13	22.6
$c\hat{m}prw$	GBM	0.054	2.05	1.49	2.05	22.9

Table VI: Statistics, in percentage, on the relative error between the mass computed by a prediction model and the mass $\hat{m}_{11,future}$ adjusted for each example trajectory, for the dataset $A320_{small}$.

type	method	mean	stdev	mean abs	rmse	max abs
A319	m_{ref}	0.402	8.11	6.3	8.12	28.5
A319	$\hat{m}_{LS}$	3.51	4.66	4.22	5.84	100
A319	$\hat{m}_{Ad}$	1.68	4.08	3.45	4.42	33.6
A319	GBM	0.508	2.54	1.87	2.59	34.1
A320	m_{ref}	0.0414	7.32	5.79	7.32	33.8
A320	$\hat{m}_{LS}$	2.49	3.08	3.1	3.97	43.4
A320	$\hat{m}_{Ad}$	1.45	3.42	2.91	3.71	26
A320	GBM	0.497	2	1.53	2.06	25.2
A321	m_{ref}	-3.72	8.64	7.88	9.4	28
A321	$\hat{m}_{LS}$	4.18	2.91	4.41	5.09	34
A321	$\hat{m}_{Ad}$	2.1	3.71	3.35	4.26	25.8
A321	GBM	0.533	2.28	1.67	2.34	24.9
A332	m_{ref}	-14.2	9.59	15.1	17.1	30.2
A332	$\hat{m}_{LS}$	-0.0847	3.9	2.66	3.9	28.7
A332	$\hat{m}_{Ad}$	-4.34	5.74	5.88	7.19	25.1
A332	GBM	0.384	2.46	1.61	2.49	25.3
B737	m_{ref}	-2.59	9.07	8.22	9.43	25.1
B737	$\hat{m}_{LS}$	8.76	10.7	9.38	13.8	75.3
B737	$\hat{m}_{Ad}$	3.77	7.53	5.84	8.42	34.7
B737	GBM	0.558	3.82	2.75	3.86	20.3
B744	m_{ref}	-22.7	6.57	22.8	23.6	39
B744	$\hat{m}_{LS}$	3.34	3.58	3.61	4.89	27.9
B744	$\hat{m}_{Ad}$	-8.23	6.26	8.74	10.3	26.8
B744	GBM	0.397	2.54	1.86	2.57	13.7
B772	m_{ref}	-16.4	6.2	16.6	17.6	30.6
B772	$\hat{m}_{LS}$	1.43	2.25	2.06	2.67	30.3
B772	$\hat{m}_{Ad}$	-3.29	4.03	4.14	5.2	16.8
B772	GBM	0.403	1.48	1.15	1.53	16.1
E145	m_{ref}	0.993	7.75	5.75	7.82	35.1
E145	$\hat{m}_{LS}$	5.68	4.42	6.13	7.2	28.6
E145	$\hat{m}_{Ad}$	3.71	4.34	4.61	5.71	23.7
E145	GBM	0.755	2.72	2.05	2.82	15.9
F100	m_{ref}	1.02	12.5	11.1	12.5	34.3
F100	$\hat{m}_{LS}$	1.92	4.8	3.95	5.16	40.6
F100	$\hat{m}_{Ad}$	1.29	5.44	4.49	5.59	22.7
F100	GBM	0.496	3.57	2.61	3.6	21.1

Table VII: Statistics, in percentage, on the relative error between the mass computed by a prediction model and the mass $\hat{m}_{11,future}$ adjusted for each example trajectory.

the document we will use this setup (GBM with $c\hat{m}prw$) and compare it with the BADA baseline and the mass estimation methods, using the 9 large sets of examples.

The results obtained for the larger datasets corresponging to 9 different aircraft types are presented in Table VII. When compared with the BADA reference mass method, the RMSE of the relative error on the mass is divided at least by 2 when using the GBM method. When compared with the two mass estimation methods, the RMSE of the relative error on the mass is reduced by at least 30 % when using the GBM method.

B. Trajectory Prediction using the Predicted Mass

In order to actually predict a trajectory using the BADA model and assuming a *max climb* thrust, one has to specify a mass, but also a speed profile. Both are usually unknown from ground systems. In our experiment, we want to evaluate the impact of the predicted mass on the trajectory prediction. Thus, we assume the speed profile to be known. The trajectory is computed using the mass predicted by the Machine Learning model and the speed profile $V_a = V_a^{(obs)}(t)$ observed on the future points. With this setup, we just look at the influence of the predicted mass – and the energy rate prediction – on the altitude prediction, disregarding the additional errors that might be induced by erroneous assumptions on the speed intent.

The results obtained on the trajectory prediction are presented in table VIII. The performance ranking of the methods is the same as in table VII. This was to be expected, as both the computation of the response variable $\hat{m}_{11,future}$ in our examples and the energy rate and altitude prediction using the predicted mass rely on the same underlying physical model. Thus, an accurate mass prediction is likely to produce an accurate altitude prediction.

Using the GBM algorithm with the set $c\hat{m}prw$, the RMSE on the altitude at a 10 minutes look-ahead time is reduced by at least 58 % when compared with the baseline obtained with the BADA reference mass m_{ref}. When compared with the two mass estimation methods, the RMSE is reduced by at least 28 % when using the GBM method.

C. Results using the BADA Speed Profile

In the previous subsection, we used the observed speed profile $V_a = V_a^{(obs)}(t)$ because we were interested in assessing the errors related to a wrong mass prediction. However, in real life, this observed speed profile is not known when the predicted trajectory is computed. Thus, the results in subsection V-B are not representative of what would be obtained in an operational context. For want of anything better one can use the speed profile specified in the BADA model.

Using this BADA speed profile, table IX presents the statistics on the error on the predicted altitude at time $t = 600$ s. Considering the RMSE (6^{th} column), we see that the performance ranking of the different methods is not as clear-cut as when the speed profile is assumed to be perfectly known. However, for all aircraft types except E145, the Machine

type	method	mean	stdev	mean abs	rmse	max abs
A319	m_{ref}	1.38	1416	1084	1416	5513
A319	$\hat{m}_{LS}$	-566	824	729	999	13084
A319	$\hat{m}_{Ad}$	-261	752	614	796	6210
A319	GBM	-57.4	484	355	488	5561
A320	m_{ref}	69.8	1345	1073	1347	5473
A320	$\hat{m}_{LS}$	-419	599	567	730	7082
A320	$\hat{m}_{Ad}$	-228	663	549	701	5755
A320	GBM	-56.2	403	303	406	5552
A321	m_{ref}	757	1633	1508	1800	5673
A321	$\hat{m}_{LS}$	-745	558	799	931	6487
A321	$\hat{m}_{Ad}$	-373	709	627	801	5481
A321	GBM	-92.9	456	333	465	5319
A332	m_{ref}	2712	1877	2876	3298	6671
A332	$\hat{m}_{LS}$	72.5	754	554	758	4139
A332	$\hat{m}_{Ad}$	829	1091	1145	1370	4813
A332	GBM	-21.7	479	308	479	3890
B737	m_{ref}	500	1767	1619	1836	4873
B737	$\hat{m}_{LS}$	-1533	1859	1669	2409	11217
B737	$\hat{m}_{Ad}$	-696	1421	1113	1582	6282
B737	GBM	-141	752	557	765	4155
B744	m_{ref}	5355	1659	5380	5606	10566
B744	$\hat{m}_{LS}$	-743	803	813	1094	5592
B744	$\hat{m}_{Ad}$	1866	1475	1987	2378	6974
B744	GBM	-112	604	439	614	3229
B772	m_{ref}	3888	1466	3915	4155	7964
B772	$\hat{m}_{LS}$	-261	508	440	571	6121
B772	$\hat{m}_{Ad}$	770	906	944	1189	4057
B772	GBM	-49.1	338	254	341	3571
E145	m_{ref}	-69.1	1847	1381	1848	7380
E145	$\hat{m}_{LS}$	-1208	1049	1355	1600	5745
E145	$\hat{m}_{Ad}$	-753	1053	1042	1294	5822
E145	GBM	-47.2	694	498	696	4781
F100	m_{ref}	-71.7	2402	2160	2403	5582
F100	$\hat{m}_{LS}$	-270	975	787	1012	6800
F100	$\hat{m}_{Ad}$	-154	1104	906	1115	4653
F100	GBM	10.1	730	535	730	4348

Table VIII: Statistics, in feet, on the difference between the predicted and observed altitudes $\left(H_p^{(pred)}(\hat{m}_{11}) - H_p^{(obs)}\right)$ at time $t = 600$ s. The trajectories are computed using the speed profile $V_a = {V_a}^{(obs)}(t)$.

Learning approach using GBM still significantly improves the altitude prediction when compared with the baseline BADA method, with benefits ranging from 29 % (F100) to about 85 % (B772 and B744), in percentage. For the E145 type, we see that neither the GBM method nor the mass estimation methods improve the results.

When comparing GBM with the mass estimation methods, disregarding the E145 case, we see that GBM performs better, with a benefit of at least 17 %, in all cases except for type F100 where the performances are similar.

The reason why the performance ranking of the different methods changes when taking the default BADA speed intent instead of the observed speed profile is quite simple. When investigating our results, we found out huge differences between the actual speed profile and the BADA speed profile, especially for the aircraft types E145, F100, and B744. The root mean square of $\left({V_a}^{(BADA)}({H_p}^{(obs)}) - {V_a}^{(obs)}\right)$ for the aircraft E145, F100 and B744 are respectively 76 kts, 42 kts and 36 kts while it is around 20 kts for the other aircraft types. The prediction for the B744 is still significantly improved when compared with the reference mass method because this reference mass is very different from the mass $\hat{m}_{11,future}$. For the two other aircraft types for which ${V_a}^{(BADA)}$ is not in accordance with the observed speed ${V_a}^{(obs)}$, the benefit of learning or estimating the mass is pretty much reduced, because of this poor estimation of the speed profile.

Note that if we focus on the 6 aircraft types for which the speed profile is relatively correct, the RMSE reduction ranges from 46 % to 86 % when comparing the Machine Learning approach with the BADA baseline, and from 17 % to 49 % when comparing with the mass estimation methods. In future work, the results might be improved by learning a model computing the future speed profile from the available variables.

type	method	mean	stdev	mean abs	rmse	max abs
A319	m_{ref}	274	1472	1176	1497	5315
A319	$\hat{m}_{LS}$	-287	1091	856	1129	11523
A319	$\hat{m}_{Ad}$	24.9	991	775	991	5749
A319	GBM	237	772	605	808	5350
A320	m_{ref}	290	1420	1165	1449	5753
A320	$\hat{m}_{LS}$	-187	876	683	896	6910
A320	$\hat{m}_{Ad}$	7.8	901	697	902	6525
A320	GBM	187	715	553	739	6815
A321	m_{ref}	863	1683	1588	1891	6154
A321	$\hat{m}_{LS}$	-646	878	883	1090	5904
A321	$\hat{m}_{Ad}$	-262	952	766	987	5225
A321	GBM	33.1	783	571	784	4627
A332	m_{ref}	2622	1820	2783	3192	6769
A332	$\hat{m}_{LS}$	-13.8	862	619	862	5568
A332	$\hat{m}_{Ad}$	729	1109	1096	1327	5622
A332	GBM	-107	673	479	682	5217
B737	m_{ref}	606	1750	1619	1852	4157
B737	$\hat{m}_{LS}$	-1480	1930	1680	2432	11105
B737	$\hat{m}_{Ad}$	-608	1451	1138	1572	6310
B737	GBM	-40.8	796	616	797	3672
B744	m_{ref}	5558	1646	5580	5797	10183
B744	$\hat{m}_{LS}$	-622	1056	950	1226	5826
B744	$\hat{m}_{Ad}$	2014	1459	2131	2487	6159
B744	GBM	12.4	844	649	844	3495
B772	m_{ref}	3728	1413	3750	3987	7145
B772	$\hat{m}_{LS}$	-280	693	580	747	5551
B772	$\hat{m}_{Ad}$	618	940	905	1125	3400
B772	GBM	-80.2	534	425	540	3513
E145	m_{ref}	1623	1801	1909	2425	7428
E145	$\hat{m}_{LS}$	460	2462	1976	2504	12908
E145	$\hat{m}_{Ad}$	934	2300	1931	2482	9650
E145	GBM	1667	2064	2032	2653	8280
F100	m_{ref}	556	1879	1616	1959	6539
F100	$\hat{m}_{LS}$	383	1266	1066	1323	5558
F100	$\hat{m}_{Ad}$	494	1211	1055	1308	4736
F100	GBM	642	1229	1166	1387	4940

Table IX: These statistics, in feet, are computed on the differences between the predicted altitude and the observed altitude $\left(H_p^{(pred)}(\hat{m}_{11}) - H_p^{(obs)}\right)$ at the time $t = 600$ s. The trajectories are computed using the BADA speed profile $V_a = V_{aBADA}$.

D. Discussion on the results

The mass estimation methods rely on a physical model (BADA) and the past trajectory points to compute the prediction. When making this prediction, we implicitely assume that the thrust law for the future points is the same as for the past points (e.g. *max climb* thrust all along the climb). This assumption might not always be true, however. The pilot may change the thrust settings during the climb for number of reasons : Air Traffic Control procedures or instructions, airline procedures, noise abatement, etc. Consequently, if our

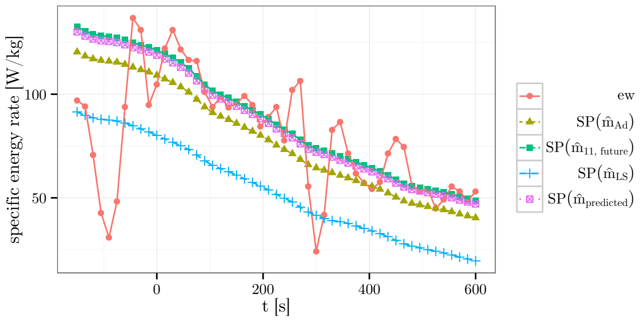

Figure 5: This figure portrays the computed specific power SP and the observed specific energy rate e_w. Only one aircraft is considered here, however different masses are used to compute the specific power SP. According Newton's law, the specific power SP of the aircraft is equal to the observed specific energy rate e_w. The past points have a negative time.

assumption on the future thrust setting is wrong, this will result in a wrong model of the specific power, as illustrated on figure 5. On this example, the observed energy rate e_w shows high variations before $t = 0$, and seems to stabilize at a higher level after $t = 0$. This is a typical case where the least square estimation method performs poorly : the modeled specific power ($SP(\hat{m}_{LS})$ on the figure) is adjusted on the points before $t = 0$, and gives a poor prediction after $t = 0$. The adaptive method performs slightly better: $SP(\hat{m}_{Ad})$ is closer to e_w on the future points. This is due to the adaptive mechanism limiting the influence of high variations of the observed energy rate.

On the example in figure 5, we see that the modeled power $SP(\hat{m}_{predicted})$ using the predicted mass fits the observed energy rate e_w quite well on the points after $t = 0$. It is also quite close to $SP(\hat{m}_{11,future})$, which is the best approximation we can make when using the mass adjusted on the future points.

In all methods, the mass is computed from the data that is available at $t = 0$ *i.e.* at the moment when the trajectory prediction would be computed in an operational context. However, the Machine Learning models make use of far more variables than the mass estimation methods. For instance, Machine Learning methods can use the distance between the departure and arrival airports as an explanatory variable. Such data is irrelevant in a physical model of the forces, though it does bring some information on the actual mass: the take-off weight of an aircraft depends on the fuel necessary to cross the distance to go. The Machine Learning approach can make use of such information, whereas the mass estimation methods cannot.

Conclusion

To conclude, let us summarize our approach and findings, before giving a few perspectives on future works. In this study, we have tested Machine Learning methods using real Mode-C radar data. A model predicting the aircraft mass from a vector of explanatory variables is learned using Machine Learning techniques. This model is learned from a set of examples in which the response variable $\hat{m}_{11,future}$ is extracted from the "future" points of each example trajectory.

We have compared our Machine Learning approach with the baseline Eurocontrol BADA (Base of Aircraft Data) method and with two state-of-the-art methods where the mass is estimated using the past trajectory points only. This comparison is made using a 10-fold cross-validation and for nine different aircraft types departing from the two main airports in Paris area.

In a first step, we have assumed the "future" speed profile to be perfectly known so as to assess only the influence of the mass accuracy on the altitude prediction. When comparing our Machine Learning approach with the baseline, the RMSE on the predicted altitude is reduced by at least 58 %, and up to nearly 92 %, depending on the aircraft type. When comparing with the mass estimation methods, the RMSE reduction when using our method ranges from about 28 % to 52 %

In a second step, we have used the default BADA speed profile so as to be as close as possible to the operational context where the actual speed intent is not known. In this context, the benefit of learning or estimating the mass is greatly reduced when the default BADA speed profile is far from the actual speed profile, which was the case for three aircraft types out of nine in our experimental setup. For the 6 remaining aircraft types, the reduction in RMSE on the altitude prediction ranges from 46 % to 86 % when comparing the Machine Learning approach with the BADA baseline, and from 17 % to 49 % when comparing with the mass estimation methods.

Concerning the 3 aircraft types (F100, E145, B744) for which the BADA speed profile poorly approximates the actual speed profile, the results still show a 85 % reduction in RMSE for the B744 type when compared with the baseline, and a 28 % reduction when comparing with the mass estimation methods. This is because the actual mass for this aircraft (Boeing 747-400) is also very poorly approximated by the BADA reference mass: Improving the mass prediction for such a heavy aircraft still does improve the altitude prediction, even when the speed profile is poorly estimated. For the F100 type (Fokker 100), the Machine Learning approach and the mass estimation methods give similar results, with a reduction in RMSE around 30 %. Finally, there is only one aircraft type (E145) for which the poor BADA approximation of the speed profile cancels the benefits of learning or estimating the mass.

In future work, these results might be improved again by learning a model predicting the future speed profile from the explanatory variables.

From an operational point of view, the resulting improvement in the climb prediction accuracy would certainly benefit air traffic controllers, especially in the vertical separation task as shown in [17]. In future works, it could be interesting to test this method on Mode-S radar data which are more accurate than Mode-C radar data, and to extend our study to other airports.

References

[1] SESAR Consortium. Milestone Deliverable D3: The ATM Target Concept. Technical report, 2007.

[2] H. Swenson, R. Barhydt, and M. Landis. Next Generation Air Transportation System (NGATS) Air Traffic Management (ATM)-Airspace Project. Technical report, National Aeronautics and Space Administration, 2006.

[3] X. Prats, V. Puig, J. Quevedo, and F. Nejjari. Multi-objective optimisation for aircraft departure trajectories minimising noise annoyance. *Transportation Research Part C*, 18(6):975–989, 2010.

[4] G. Chaloulos, E. Crück, and J. Lygeros. A simulation based study of subliminal control for air traffic management. *Transportation Research Part C*, 18(6):963–974, 2010.

[5] James K Kuchar and Lee C Yang. A review of conflict detection and resolution modeling methods. *Intelligent Transportation Systems, IEEE Transactions on*, 1(4):179–189, 2000.

[6] Lucia Pallottino, Eric M Feron, and Antonio Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *Intelligent Transportation Systems, IEEE Transactions on*, 3(1):3–11, 2002.

[7] J. M. Alliot, Hervé Gruber, and Marc Schoenauer. Genetic algorithms for solving ATC conflicts. In *Proceedings of the Ninth Conference on Artificial Intelligence Application*. IEEE, 1992.

[8] N. Durand, J.M. Alliot, and J. Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.

[9] Nicolas Durand and Jean-Marc Alliot. Ant colony optimization for air traffic conflict resolution. In *8th USA/Europe Air Traffic Management Research and Developpment Seminar*, 2009.

[10] C. Vanaret, D. Gianazza, N. Durand, and J.B. Gotteland. Benchmark for conflict resolution (regular paper). In *International Conference on Research in Air Transportation (ICRAT), Berkeley, California, 22/05/12-25/05/12*, pages 1–8, http://www.icrat.org, may 2012. ICRAT.

[11] G Mykoniatis and P Martin. Study of the acquisition of data from aircraft operators to aid trajectory prediction calculation. Technical report, EUROCONTROL Experimental Center, 1998.

[12] ADAPT2. aircraft data aiming at predicting the trajectory. data analysis report. Technical report, EUROCONTROL Experimental Center, 2009.

[13] R. A. Coppenbarger. Climb trajectory prediction enhancement using airline flight-planning information. In *AIAA Guidance, Navigation, and Control Conference*, 1999.

[14] J. Lopez-Leones, M.A. Vilaplana, E. Gallo, F.A. Navarro, and C. Querejeta. The aircraft intent description language: A key enabler for air-ground synchronization in trajectory-based operations. In *Proceedings of the 26th IEEE/AIAA Digital Avionics Systems Conference*. DASC, 2007.

[15] Javier Lopez Leones. *Definition of an aircraft intent description language for air traffic management applications*. PhD thesis, University of Glasgow - Department of Aerospace Sciences, 2008.

[16] Richard Alligier, David Gianazza, and Nicolas Durand. Ground-based estimation of aircraft mass, adaptive vs. least squares method. In *10th USA/Europe Air Traffic Management Research and Developpment Seminar*, 2013.

[17] C. Schultz, D. Thipphavong, and H. Erzberger. Adaptive trajectory prediction algorithm for climbing flights. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, August 2012.

[18] A.W. Warren and Y.S. Ebrahimi. Vertical path trajectory prediction for next generation atm. In *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE*, volume 2, pages F11/1 –F11/8 vol.2, oct-7 nov 1998.

[19] A.W. Warren. Trajectory prediction concepts for next generation air traffic management. In *3rd USA/Europe ATM R&D Seminar*, June 2000.

[20] G. L. Slater. Adaptive improvement of aircraft climb performance for air traffic control applications. In *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*. IEEE conference publications, October 2002.

[21] I. Lymperopoulos, J. Lygeros, and A. Lecchini Visintini. Model Based Aircraft Trajectory Prediction during Takeoff. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 1–12, Keystone, Colorado, August 2006.

[22] R. Alligier, D. Gianazza, and N. Durand. Energy Rate Prediction Using an Equivalent Thrust Setting Profile (regular paper). In *International Conference on Research in Air Transportation (ICRAT), Berkeley, California, 22/05/12-25/05/12*, pages 1–7, http://www.icrat.org, may 2012. ICRAT.

[23] R. Alligier, D. Gianazza, and N. Durand. Learning the aircraft mass and thrust to improve the ground-based trajectory prediction of climbing flights. *Transportation Research Part C: Emerging Technologies*, 36(0):45 – 60, 2013.

[24] A. Hadjaz, , G. Marceau, P. Savéant, and M. Schoenauer. Online learning for ground trajectory prediction. In *Proceedings of the SESAR Innovation Days (2012)*. EUROCONTROL, 2012.

[25] David P Thipphavong, Charles A Schultz, Alan G Lee, and Steven H Chan. Adaptive algorithm to improve trajectory prediction accuracy of climbing aircraft. *Journal of Guidance, Control, and Dynamics*, 36(1):15–24, 2012.

[26] Young S. Park and David P. Thipphavong. Performance of an Adaptive Trajectory Prediction Algorithm for Climbing Aircraft. In *2013 Aviation Technology, Integration, and Operations Conference*, page (on line). 08, Aug 2013.

[27] Richard Alligier, David Gianazza, Mohammad Ghasemi Hamed, and Nicolas Durand. Comparison of Two Ground-based Mass Estimation Methods on Real Data (regular paper). In *International Conference on Research in Air Transportation (ICRAT), Istambul, 26/05/2014-30/05/2014*, pages 1–8, http://www.icrat.org, mai 2014. ICRAT.

[28] A. Nuic. User manual for base of aircarft data (bada) rev.3.9. Technical report, EUROCONTROL, 2011.

[29] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk minimization. *Pattern Recogn. Image Anal.*, 1(3):284–305, 1991.

[30] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[31] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208. ACM, 1999.

[32] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143. Morgan Kaufmann, 1995.

[33] C. R. Rao and H. Toutenburg. *Linear Models: Least Squares and Alternatives (Springer Series in Statistics)*. Springer, July 1999.

[34] Alan Miller. *Subset selection in regression*. CRC Press, 2002.

[35] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[36] Ian T. Jolliffe. A note on the use of principal components in regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(3):pp. 300–303, 1982.

[37] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[38] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38(4):367 – 378, 2002.

[39] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.

Richard Alligier received his Ph.D. (2014) degree in Computer Science from the "Institut National Polytechnique de Toulouse" (INPT), his engineer's degrees (IEEAC, 2010) from the french university of civil aviation (ENAC) and his M.Sc. (2010) in computer science from the University of Toulouse. He is currently assistant professor at the ENAC in Toulouse, France.

David Gianazza received his two engineer degrees (1986, 1996) from the french university of civil aviation (ENAC) and his M.Sc. (1996) and Ph.D. (2004) in Computer Science from the "Institut National Polytechnique de Toulouse" (INPT). He has held various positions in the french civil aviation administration, successively as an engineer in ATC operations, technical manager, and researcher. He is currently associate professor at the ENAC, Toulouse.

Nicolas Durand graduated from the Ecole polytechnique de Paris in 1990 and the Ecole Nationale de l'Aviation Civile (ENAC) in 1992. He has been a design engineer at the Centre d'Etudes de la Navigation Aérienne (then DSNA/DTI R&D) since 1992, holds a Ph.D. in Computer Science (1996) and got his HDR (french equivalent of tenure) in 2004 both from the "Institut National Polytechnique de Toulouse" (INPT). He is currently professor at the ENAC/MAIAA lab.